

CLAIMS

1. (Previously Presented) A method for providing quality of service for applications in multiple transport protocol environments which comprises:
 - creating a QoS negotiation request for a client application at a client QoS negotiator;
 - transmitting the QoS negotiation request from the client QoS negotiator to a server QoS negotiator;
 - adjusting server QoS parameters in response to the QoS negotiation request;
 - creating a QoS negotiation response at the server QoS negotiator, the QoS negotiation response containing connection information and server QoS information;
 - transmitting the QoS negotiation response to the client QoS negotiator;
 - adjusting client QoS parameters in response to the QoS negotiation response; and
 - connecting the client application to a server application using the connection information and the server QoS information.

2. (Original) The method of claim 1 further comprising:
 - monitoring the client QoS parameters and the server QoS parameters as the client application and the server application communicate;
 - detecting changes in network conditions and data requirements of the client application and the server application; and
 - adjusting the client QoS parameters and the server QoS parameters in response to said changes.

3. (Original) The method of claim 2 wherein the step of adjusting server QoS parameters further comprises adjusting server bandwidth, server buffer, and server cache parameters.

4. (Original) The method of claim 3 wherein the step of adjusting client QoS parameters further comprises adjusting client bandwidth, client buffer, and client cache parameters.

5. (Previously Presented) A method for providing dynamic profile management for a client which comprises:
receiving an application profile request from a client application;
constructing a QoS request for the client application based at least in part on the application profile;
transmitting the QoS request to a server;
receiving a QoS response from the server;
adjusting client settings based upon the QoS response; and
connecting the client application to a server application residing at the server using connection information and server QoS information stored in the QoS response.

6. (Original) The method of claim 5 further comprising:
monitoring the client application for changes in data requirements;
detecting changes in network conditions at the client;
sending a second QoS request to the server in response to the changes in data requirements or the changes in network conditions;
receiving a second QoS response from the server; and
adjusting the client parameters in response to the second QoS response.

7. (Original) The method of claim 6 further comprising repeating the steps of claim 6 until execution of the client application terminates.

8. (Original) The method of claim 7 wherein the step of constructing the QoS request further comprises:

identifying application type information and application QoS requirements;
and
storing the application type information and application Qos requirements
in the QoS request.

9. (Original) The method of claim 8 wherein the step of adjusting
client settings further comprises setting bandwidth, buffer, and queue parameters of the
client.

10. (Original) A method for providing dynamic profile management for
a server which comprises:

receiving a QoS request originating from a client at the server;
constructing a QoS response containing connection information and
server QoS information;
adjusting server parameters in response to the QoS request;
transmitting the QoS response to the client; and
connecting a server application residing at the server to a client
application based upon the connection information and server QoS information.

11. (Original) The method of claim 10 further comprising:
receiving a second QoS request send by the client in response to changes
in data requirements or network conditions;
adjusting server parameters in response to the second QoS request;
creating a second QoS response; and
transmitting the second QoS response to the client.

12. (Original) The method of claim 11 wherein the step of adjusting
server parameters further comprises setting bandwidth, buffer, and queue parameters
of the server.

13. (Previously Presented) A generic quality of service protocol comprising:
an ICMP header for transmitting the protocol as an out-of-band message;
a client information storage unit;
a proxy information storage unit;
an application profile information storage unit;
means for storing transport QoS profile information;
means for storing per-protocol QoS profile information; and
means for storing QoS map order information.

14. (Original) The protocol of claim 13, wherein said client information storage unit further comprises:
means for storing operating system type information;
means for storing workstation configuration information;
means for storing processor architecture information; and
means for storing network architecture information.

15. (Original) The protocol of claim 14, wherein said proxy information storage unit further comprises:
means for storing proxy IP addresses; and
means for storing proxy port numbers.

16. (Original) The protocol of claim 15, wherein said application profile information storage unit further comprises:
means for storing application source information;
means for storing application class information;
means for storing application bandwidth requirements;
means for storing data transfer rates; and
means for storing response times.

17. (Original) The protocol of claim 16, wherein said means for storing transport QoS profile information further comprises:

means for storing protocol available client protocols; and
means for storing server protocol grants.

18. (Original) The protocol of claim 17, wherein said means for storing per-protocol QoS profile information further comprises:

means for storing ATM connection information; and
means for storing ATM address information.

19. (Original) A generic quality of service architecture comprising:
a client QoS negotiator in communication with a client application;
a server QoS negotiator in communication with a server application;
a generic QoS protocol accessible by the client QoS negotiator and the server QoS negotiator; and
a generic QoS API for configuring, monitoring, and maintaining the client QoS negotiator, the server QoS negotiator, and the generic QoS protocol.

20. (Original) The architecture of claim 19, wherein said client QoS negotiator is disposed above and communicates with a client socket layer.

21. (Original) The architecture of claim 20, wherein said client socket layer further comprises ATM, RSVP, TCP/UDP, and IPv6 protocols.

22. (Previously Presented) The architecture of claim 21, wherein said server QoS negotiator is disposed above and communicates with a server socket layer.

23. (Original) The architecture of claim 22, wherein said server socket layer further comprises ATM, RSVP, TCP/UDP, and IPv6 protocols.

24. (Original) The architecture of claim 23 wherein the client QoS negotiator negotiates a QoS profile with the server QoS negotiator by exchanging messages and sharing information through the generic QoS protocol.

25. (Original) The architecture of claim 24 wherein the client QoS negotiator sets local bandwidth, buffer, and cache parameters for the client application.

26. (Previously Presented) The architecture of claim 25 wherein the server QoS negotiator sets local bandwidth, buffer, and cache parameters for the server application.

27. (Original) The architecture of claim 26 wherein the client QoS negotiator and the server QoS negotiator connect the client application to the server application based upon the QoS profile.